1. (AMENDED) A [Turing complete] computer implemented learning method comprising the steps of:

(a) providing a computer with an indirectly executable [computer] program including:

a first instruction that points to and designates machine code stored in a memory as data;

a second instruction that points to and designates machine code stored in [the] a memory as [at least one] directly executable [function] code;

a third instruction that alters machine code pointed to by the first instruction; and

a fourth instruction that executes machine code pointed to by the second instruction; and

(b) controlling the computer to execute the program [to perform] which performs the steps of:

(b1) creating and storing a machine code entity including [at least one] a directly executable [function having a directly executable branch] instruction in a memory, said directly executable instruction causing the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said directly executable instruction in the entity;

(b2) executing the second instruction to point to the entity;

(b3) executing the [third] fourth instruction using [training] input data to produce a result;

(b4) [processing] evaluating the result using a learning algorithm;

(b5) executing the first instruction to point to the entity;

(b6) [altering] executing the third instruction to alter the entity using an alteration algorithm to include [at least one other] a different directly executable [function] instruction;

and

(b7) repeating steps (b3) to (b6) until an end criterion is reached.

2. (AMENDED) A method as in claim 1, in which [the branch] said directly executable instruction is a recursive instruction.

3. (AMENDED) A method as in claim 1, in which [the branch] said directly executable instruction is a leaf function call.

4. (AMENDED) A method as in claim 1, in which [the branch] said directly executable instruction is an external function call.

5. (AMENDED) A method as in claim 1, in which [the branch] said directly executable instruction is a subroutine call.

Q1

6. (AMENDED) A method as in claim 1, in which:

the first instruction points to and designates machine code stored in a memory as data by casting a pointer [to the entity] thereto as a data pointer; and

the second instruction points to and designates machine code stored in [the] a memory as [a] directly executable [function] code by casting [said] a pointer thereto as a function pointer.

7. (AMENDED) A method as in claim 1, in which:

[said at least one directly executable function as comprises] step (b1) comprises creating the entity as comprising a return instruction; and

step (b6) comprises preventing the return instruction from being altered.

~~9~~8 (AMENDED) A method as in claim 1, in which:

step (b1) comprises creating and storing the entity as [an array of functions;

step (b2) comprises executing the second instruction to point to the array; and

step (b5) comprises executing the first instruction to point to the array] including a plurality of routines, at least one of which includes a directly executable instruction.

9 ~~10~~. (AMENDED) A method as in claim 8 ~~9~~, in which:

each [function] <u>routine</u> includes at least one directly executable instruction; and

the [functions] <u>routines</u> can include different numbers of directly executable instructions.

10 ~~11~~. (AMENDED) A method as in claim 9 ~~10~~, in which the [functions] <u>routines</u> have a maximum length.

11 ~~12~~. (AMENDED) A method as in claim 8 ~~9~~, in which at least one of the [functions includes] <u>routines is</u> a subroutine.

12 ~~13~~. (AMENDED) A method as in claim 1, in which [said at least one function] <u>the entity</u> comprises a subroutine that is called by [the branch] <u>said directly executable</u> instruction.

13 ~~14~~. (AMENDED) A method as in claim 1, in which [said at least one function] <u>the entity</u> comprises a plurality of subroutines that can call each other.

14 ~~15~~. (AMENDED) A method as in claim 1, in which:

[said at least one function] <u>the entity</u> comprises a main [function] <u>routine</u> and a plurality of subroutines which have headers respectively; and

step (b1) comprises initializing the headers such that some of the subroutines cannot call others of the subroutines.

15 ~~16~~. (AMENDED) A method as in claim 1, in which:

[said at least one function] <u>the entity</u> comprises a main [function] <u>routine</u> and a subroutine having headers respectively; and

step (b1) comprises initializing the header of the subroutine such that it can call itself and thereby perform recursion.

17 ~~16~~. (AMENDED) A [Turing complete] computer learning system comprising:

<u>a</u> memory [means] for storing an indirectly executable computer program including:

a first instruction that points to and designates machine code stored in [a] the memory as data;

a second instruction that points to and designates machine code stored in the memory as [at least one] directly executable [function] machine code;

a third instruction that alters machine code pointed to by the first instruction; and

a fourth instruction that executes machine code pointed to by the second instruction; and

[processing means] a processor for executing the program;

the [processing means] processor, memory [means] and program operating in combination for performing the steps of:

(a) creating and storing a machine code entity including [at least one] a directly executable [function having a directly executable branch] instruction in the memory [means], said directly executable instruction causing the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said directly executable instruction in the entity;

[illegible]

(b) executing the second instruction to point to the entity;

(c) executing the [third] fourth instruction using [training] input data to produce a result;

(d) [processing] evaluating the result using a learning algorithm;

(e) executing the first instruction to point to the entity;

(f) [altering] executing the third instruction to alter the entity using an alteration algorithm to include [at least one other] a different directly executable [function] instruction; and

(g) repeating steps (b) to (f) until an end criterion is reached.

[illegible]

52 ~~21~~ ~~22~~. (AMENDED) A system as in claim 47 ~~17~~, in which:

the first instruction points to and designates machine code stored in [a] the memory as data by casting a pointer [to the entity] thereto as a data pointer; and

the second instruction points to and designates machine code stored in the memory as [a] directly executable [function] code by casting [said] a pointer thereto as a function pointer.

53 ~~22~~ ~~23~~. (AMENDED) A system as in claim 47 ~~17~~, in which:

[said at least one directly executable function as comprises] step (a) comprises creating the entity as comprising a return instruction; and

step [(b6)] (f) comprises preventing the return instruction from being altered.

54 ~~23~~ ~~24~~. (AMENDED) A system as in claim 47 ~~17~~, in which:

A2

step [(b1)] (a) comprises creating and storing the entity as [an array of directly executable functions;

step (b2) comprises executing the second instruction to point to the array; and

step (b5) comprises executing the first instruction to point to the array] including a plurality of routines, at least one of which includes a directly executable instruction.

55 ~~24~~ ~~25~~. (AMENDED) A system as in claim 54 ~~23~~ ~~24~~, in which:

each [function] routine includes at least one directly executable instruction; and

the [functions] routines can include different numbers of directly executable instructions.

56 ~~25~~ ~~26~~. (AMENDED) A system as in claim 55 ~~24~~ ~~25~~, in which the [functions] routines have a maximum length.

57 ~~26~~ ~~27~~. (AMENDED) A system as in claim 54 ~~23~~ ~~24~~, in which at least one of the [functions includes] routines is a subroutine.

58 ~~27~~ ~~28~~. (AMENDED) A system as in claim 47 ~~17~~, in which [said at least one function] the entity comprises a subroutine that is called by [the branch] said directly executable instruction.

81

Rule 1.126

59 ~~28~~ ~~29~~. (AMENDED) A system as in claim 47 ~~16~~ ~~17~~, in which [said at least one function] the entity comprises a plurality of subroutines that can call each other.

60 ~~29~~ ~~30~~. (AMENDED) A system as in claim 47 ~~16~~ ~~17~~, in which:

[said at least one function] the entity comprises a main [function] routine and a plurality of subroutines which have headers respectively; and

step [(b1)] (a) comprises initializing the headers such that some of the subroutines cannot call others of the subroutines.

61 ~~30~~ ~~31~~. (AMENDED) A system as in claim 47 ~~16~~ ~~17~~ in which:

[said at least one function] the entity comprises a main [function] routine and a subroutine having headers respectively; and

Q2

step [(b1)] (a) comprises initializing the header of the subroutine such that it can call itself and thereby perform recursion.

62 ~~31~~ ~~32~~. (AMENDED) A system as in claim 47 ~~16~~ ~~17~~, in which:

the memory [means] comprises a main memory and a processor memory that is part of the processor; and

step (a) comprises storing the entity in the processor memory.

63 ~~32~~ ~~33~~. (UNCHANGED) A system as in claim 62 ~~31~~ ~~32~~, comprising an integrated circuit chip, in which the processor and the processor memory are formed on the chip.

64 ~~33~~ ~~34~~. (UNCHANGED) A system as in claim 63 ~~32~~ ~~33~~, in which at least a portion of the program is stored processor memory.

65 ~~34~~ ~~35~~. (AMENDED) A [method] system as in claim 64 ~~33~~ ~~34~~, in which:

the processor memory comprises a non-volatile memory section; and

said at least a portion of the program is stored in the non-volatile memory section.

Q3

116 ~~35~~ ~~36~~. A method as in claim 1, further comprising the step, performed between steps (b1) and (b7), of:

(b8) making a copy of the entity.

17 ~~36~~ ~~37~~. A method as in claim 1, in which said directly executable instruction comprises a branch instruction.

18 ~~37~~ ~~38~~. A method as in claim 1, in which said directly executable instruction comprises a call instruction.

19 ~~38~~ ~~39~~. A method as in claim 1, in which said directly executable instruction comprises a subroutine call instruction.

20 ~~39~~ ~~40~~. A method as in claim 1, in which said directly executable instruction comprises a jump instruction.

21 ~~40~~ ~~41~~. A method as in claim 1, in which said directly executable instruction comprises a conditional jump instruction.

22 ~~41~~ ~~42~~. A method as in claim 1, in which said directly executable instruction comprises a function call.

RWD

Q3

23 ~~42~~ ~~43~~. A method as in claim 22 ~~42~~, 22 in ~~in~~ which the contents of the function call are learned by executing steps (b1) to (b7).

24 ~~43~~ ~~44~~. A method as in claim 22 ~~42~~, 22 in which in which the contents of the function call are not learned by executing steps (b1) to (b7).

25 ~~44~~ ~~45~~. A method as in claim 1, in which said directly executable instruction comprises an external function call.

26 ~~45~~ ~~46~~. A method as in claim 1, in which said directly executable instruction causes the processor to execute a procedure in which the entire state of the processor is not saved before execution of the procedure.

27 ~~46~~ ~~47~~. A method as in claim 1, in which said directly executable instruction causes the processor to execute a leaf procedure.

RWD

28 ~~47~~ ~~48~~. A method as in claim 27 ~~47~~, 27 ~~in~~ in which the contents of the leaf procedure are learned by executing steps (b1) to (b7).

29 ~~48~~ ~~49~~. A method as in claim 27 ~~47~~, 27 in which in which the contents of the leaf procedure are not learned by executing steps (b1) to (b7).

30 ~~49~~ ~~50~~. A method as in claim 1, in which said directly executable instruction causes data to be retrieved from the memory.

31 ~~50~~ ~~51.~~ A method as in claim 1, in which said directly executable instruction causes data to be retrieved from the memory using an index.

32 ~~51~~ ~~52.~~ A method as in claim 1, in which said directly executable instruction causes the processor to execute an instruction which precedes said directly executable instruction.

33 ~~52~~ ~~53.~~ A method as in claim 1, in which said directly executable instruction comprises a loop instruction.

34 ~~53~~ ~~54.~~ A method as in claim 1, in which said directly executable instruction causes the computer to execute in a recursive manner.

35 ~~54~~ ~~55.~~ A method as in claim 1, in which said directly executable instruction comprises a list instruction.

C3 36 ~~55~~ ~~56.~~ A method as in claim 1, in which said directly executable instruction comprises a string instruction.

37 ~~56~~ ~~57.~~ A method as in claim 1, in which said directly executable instruction causes the processor to link the entity to a function which is external of the entity.

38 ~~57~~ ~~58.~~ A method as in claim 1, in which said directly executable instruction comprises a protected division function.

39 ~~58~~ ~~59.~~ A method as in claim 1, in which said directly executable instruction is a logarithmic function.

40 ~~59~~ ~~60.~~ A method as in claim 1, in which said directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said instruction which directly follows said directly executable instruction in the entity.

41 ~~60~~ ~~61.~~ A method as in claim 1, in which said directly executable instruction causes data to be written to the memory.

42 ~~61~~ ~~62.~~ A method as in claim 1, in which said directly executable instruction causes data to be written to the memory using an index.

43 ~~62~~ ~~63.~~ A method as in claim 1, in which said directly executable instruction causes data in the memory to be altered.

84

44 ~~63~~ ~~64~~. A method as in claim 1, in which said directly executable instruction a causes data in the memory to be altered using an index.

45 ~~64~~ ~~65~~. A method as in claim 1, in which said directly executable instruction comprises a control transfer instruction.

46 ~~65~~ ~~66~~. A method as in claim 1, in which:

the entity comprises a main routine and a plurality of subroutines; and

step (b1) comprises creating and storing the entity such that some of the subroutines cannot call others of the subroutines.

66 ~~67~~. A method as in claim 47 ~~17~~ 47, further comprising the step, performed between steps (b1) and (b7), of:

(b8) making a copy of the entity.

67 ~~68~~. A system as in claim 47 ~~17~~ 47, in which said directly executable instruction comprises a branch instruction.

68 ~~69~~. A system as in claim 47 ~~17~~ 47 in which said directly executable instruction comprises a call instruction.

69 ~~70~~. A system as in claim 47 ~~17~~ 47, in which said directly executable instruction comprises a subroutine call instruction.

70 ~~71~~. A system as in claim 47 ~~17~~ 47, in which said directly executable instruction comprises a jump instruction.

71 ~~72~~. A system as in claim 47 ~~17~~ 47 in which said directly executable instruction comprises a conditional jump instruction.

72 ~~73~~. A system as in claim 47 ~~17~~ 47, in which said directly executable instruction comprises a function call.

73 ~~74~~. A system as in claim 72 ~~73~~, ~~in~~ in which the contents of the function call are learned by executing steps (b1) to (b7).

74 ~~75~~. A system as in claim 72 ~~73~~, in which in which the contents of the function call are not learned by executing steps (b1) to (b7).

75 ~~76~~. A system as in claim 47 ~~17~~ 47, in which said directly executable instruction comprises an external function call.

D3

RWD

85

~~77~~ 76. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the processor to execute a procedure in which the entire state of the processor is not saved before execution of the procedure.

~~78~~ 77. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the processor to execute a leaf procedure.

RWD

~~79~~ 78. A system as in claim ~~78~~ 77, ~~in~~ in which the contents of the leaf procedure are learned by executing steps (b1) to (b7).

~~80~~ 79. A system as in claim ~~78~~ 77, in which in which the contents of the leaf procedure are not learned by executing steps (b1) to (b7).

Q3

~~81~~ 80. A system as in claim ~~17~~ 47, in which said directly executable instruction causes data to be retrieved from the memory.

~~82~~ 81. A system as in claim ~~17~~ 47, in which said directly executable instruction causes data to be retrieved from the memory using an index.

~~83~~ 82. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the processor to execute an instruction which precedes said directly executable instruction.

~~84~~ 83. A system as in claim ~~17~~ 47, in which said directly executable instruction comprises a loop instruction.

~~85~~ 84. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the computer to execute in a recursive manner.

~~86~~ 85. A system as in claim ~~17~~ 47, in which said directly executable instruction comprises a list instruction.

~~87~~ 86. A system as in claim ~~17~~ 47, in which said directly executable instruction comprises a string instruction.

~~88~~ 87. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the processor to link the entity to a function which is external of the entity.

[illegible]

~~89~~ 88. A system as in claim ~~17~~ 47, in which said directly executable instruction comprises a protected division function.

~~90~~ 89. A system as in claim ~~17~~ 47, in which said directly executable instruction is a logarithmic function.

~~91~~ 90. A system as in claim ~~17~~ 47, in which said directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said instruction which directly follows said directly executable instruction in the entity.

~~92~~ 91. A system as in claim ~~17~~ 47, in which said directly executable instruction causes data to be written to the memory.

D3

~~93~~ 92. A system as in claim ~~17~~ 47, in which said directly executable instruction causes data to be written to the memory using an index.

~~94~~ 93. A system as in claim ~~17~~ 47, in which said directly executable instruction causes data in the memory to be altered.

~~95~~ 94. A system as in claim ~~17~~ 47 in which said directly executable instruction a causes data in the memory to be altered using an index.

~~96~~ 95. A system as in claim ~~17~~ 47, in which said directly executable instruction comprises a control transfer instruction.

Sub B' ~~97~~ 96. A computer implemented learning method using a digital computer which includes a processor and a memory, comprising the steps of:

(a) selecting a directly executable instruction;

(b) determining a binary number which represents the directly executable instruction to the processor;

(c) selecting a location in the memory;

(d) creating a directly executable machine code entity which includes the binary number;

(e) storing the entity as data in said location;

(f) providing an indirectly executable instruction which causes the processor to directly execute machine code stored in

[illegible]

said location;

(g) controlling the processor to process the indirectly executable instruction, resulting in the processor directly executing the entity as including the directly executable instruction;

(h) altering the entity as data such that the binary number is changed to represent a different directly executable machine code program instruction to the processor; and

(i) repeating steps (c) to (h) until an end criterion is reached.

~~98~~ 97. A method as in claim ~~97~~ 96, further comprising the step, performed before step (a), of:

(j) allocating space in the memory for the entity such that said space will remain allocated for a plurality of executions of steps (c) to (h).

~~99~~ 98. A method as in claim ~~97~~ 96, in which steps (a) to (e) comprise creating and storing the entity as including binary numbers representing a plurality of directly executable instructions respectfully which are arranged to emulate a tree structure.

~~100~~ 99. A method as in claim ~~99~~ 98, in which some of the directly executable instructions are arranged to constitute parenthesis in the tree structure.

~~101~~ 100. A method as in claim ~~97~~ 96, in which steps (a) to (e) comprise creating and storing the entity as including binary numbers representing a plurality of directly executable instructions respectively, a group of which is protected from being broken apart while performing step (h).

~~102~~ 101. A method as in claim ~~101~~ 100, in which steps (a) to (e) further comprise creating and storing the entity as including protection data that indicates boundaries of said group for protecting said group from being broken apart.

102 ~~103~~. A method as in claim 100 ~~101~~, in which steps (a) to (e) further comprises storing protection data separate from the entity, the protection data indicating boundaries of said group for protecting said group from being broken apart.

103 ~~104~~. A method as in claim 100 ~~101~~, in which step (h) comprises performing genetic crossover.

104 ~~105~~. A method as in claim 100 ~~101~~, in which step (h) comprises performing genetic mutation.

105 ~~106~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said directly executable instruction in the entity.

106 ~~107~~. A method as in claim 96 ~~97~~, further comprising the step, performed between steps (d) and (h), of:

(b8) making a copy of the entity.

107 ~~108~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a branch instruction.

108 ~~109~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a call instruction.

109 ~~110~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a subroutine call instruction.

110 ~~111~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a jump instruction.

111 ~~112~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a conditional jump instruction.

112 ~~113~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a function call.

RWJ

113 ~~114~~. A method as in claim 112 ~~113~~, ~~in~~ in which the contents of the function call are learned by executing steps (b1) to (b6).

114 ~~115~~. A method as in claim 112 ~~113~~, in which in which the contents of the function call are not learned by executing steps (b1) to (b6).

RWD

115 ~~116~~. A method as in claim 112 ~~113~~, ~~in~~ in which the contents of the function call are learned by executing steps (b1) to (b6).

116 ~~117~~. A method as in claim 112 ~~113~~, in which in which the contents of the function call are not learned by executing steps (b1) to (b6).

117 ~~118~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises an external function call.

118 ~~119~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to execute a procedure in which the entire state of the processor is not saved before execution of the procedure.

Q3

119 ~~120~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to execute a leaf procedure.

RWD

120 ~~121~~. A method as in claim 119 ~~120~~, ~~in~~ in which the contents of the leaf procedure are learned by executing steps (b1) to (b6).

121 ~~122~~. A method as in claim 119 ~~120~~, in which in which the contents of the leaf procedure are not learned by executing steps (b1) to (b6).

RWD

122 ~~123~~. A method as in claim 119 ~~120~~, ~~in~~ in which the contents of the leaf procedure are learned by executing steps (b1) to (b6).

123 ~~124~~. A method as in claim 119 ~~120~~, in which in which the contents of the leaf procedure are not learned by executing steps (b1) to (b6).

124 ~~125~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes data to be retrieved from the memory.

125 ~~126~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes data to be retrieved from the memory using an index.

126 ~~127~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to execute an instruction which precedes said directly executable instruction.

127 ~~128~~. A method as in claim 96 ~~97~~, in which said directly

40

executable instruction comprises a loop instruction.

128 ~~129~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the computer to execute in a recursive manner.

129 ~~130~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a list instruction.

130 ~~131~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a string instruction.

131 ~~132~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to link the entity to a function which is external of the entity.

Q3

132 ~~133~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a protected division function.

133 ~~134~~. A method as in claim 96 ~~97~~, in which said directly executable instruction is a logarithmic function.

134 ~~135~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said instruction which directly follows said directly executable instruction in the entity.

135 ~~136~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes data to be written to the memory.

136 ~~137~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes data to be written to the memory using an index.

137 ~~138~~. A method as in claim 96 ~~97~~, in which said directly executable instruction causes data in the memory to be altered.

138 ~~139~~. A method as in claim 96 ~~97~~, in which said directly executable instruction a causes data in the memory to be altered using an index.

139 ~~140~~. A method as in claim 96 ~~97~~, in which said directly executable instruction comprises a control transfer instruction.

140 ~~141~~. A method as in claim 96 ~~97~~, in which:

step (b1) comprises creating the entity as comprising a return instruction; and

step (b5) comprises preventing the return instruction from being altered.

~~142~~141. A method as in claim ~~97~~96, in which said different different directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said different different directly executable instruction in the entity.

~~143~~142. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a branch instruction.

~~144~~143. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a call instruction.

Q3

~~145~~144. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a subroutine call instruction.

~~146~~145. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a jump instruction.

~~147~~146. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a conditional jump instruction.

~~148~~147. A method as in claim ~~97~~96, in which said different directly executable instruction comprises a function call.

~~149~~148. A method as in claim ~~97~~96, in which said different directly executable instruction comprises an external function call.

~~150~~149. A method as in claim ~~97~~96, in which said different directly executable instruction causes the processor to execute a procedure in which the entire state of the processor is not saved before execution of the procedure.

~~151~~150. A method as in claim ~~97~~96, in which said different directly executable instruction causes the processor to execute a leaf procedure.

~~152~~151. A method as in claim ~~97~~96, in which said different directly executable instruction causes data to be retrieved from the memory.

~~153~~152. A method as in claim ~~97~~96, in which said different directly

92

executable instruction causes data to be retrieved from the memory using an index.

153 ~~154~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes the processor to execute an instruction which precedes said different directly executable instruction.

154 ~~155~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction comprises a loop instruction.

155 ~~156~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes the computer to execute in a recursive manner.

Q3

156 ~~157~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction comprises a list instruction.

157 ~~158~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction comprises a string instruction.

158 ~~159~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes the processor to link the entity to a function which is external of the entity.

159 ~~160~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction comprises a protected division function.

160 ~~161~~. A method as in claim 96 ~~97~~, in which said directly executable instruction is a logarithmic function.

161 ~~162~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said instruction which directly follows said different directly executable instruction in the entity.

162 ~~163~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes data to be written to the memory.

163 ~~164~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes data to be written to the memory using an index.

93

164 ~~165~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction causes data in the memory to be altered.

165 ~~166~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction a causes data in the memory to be altered using an index.

166 ~~167~~. A method as in claim 96 ~~97~~, in which said different directly executable instruction comprises a control transfer instruction.

167 ~~168~~. A method of creating and executing machine code using a digital computer which includes a processor and a memory, comprising the steps of:

(a) selecting a directly executable instruction which causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said directly executable instruction;

(b) determining a binary number which represents the directly executable instruction to the processor;

(c) selecting a location in the memory;

(d) creating a directly executable machine code entity which includes the binary number;

(e) storing the entity as data in said location;

(f) providing an indirectly executable instruction which causes the processor to directly execute machine code stored in said location; and

(g) controlling the processor to process the indirectly executable instruction, resulting in the processor directly executing the entity as including the directly executable instruction.

168 ~~169~~. A method as in claim 167 ~~168~~, further comprising the steps, performed after step (g), of:

(h) copying the entity to produce a copied entity; and

(i) altering the copied entity as data such that the

Q3

94

binary number is changed to represent a different directly executable instruction.

169 ~~170~~. A method as in claim 168 ~~169~~, in which:

step (d) comprises creating the entity as comprising a return instruction; and

step (i) comprises preventing the return instruction from being altered.

170 ~~171~~. A method as in claim 167 ~~168~~, further comprising the steps, performed after step (g), of:

(h) copying the entity to produce a copied entity; and

(i) altering the entity as data such that the binary number is changed to represent a different directly executable machine code program instruction to the processor.

a3

171 ~~172~~. A method as in claim 170 ~~171~~, in which:

step (d) comprises creating the entity as comprising a return instruction; and

step (i) comprises preventing the return instruction from being altered.

172 ~~173~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a branch instruction.

173 ~~174~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a call instruction.

174 ~~175~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a subroutine call instruction.

175 ~~176~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a jump instruction.

176 ~~177~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a conditional jump instruction.

177 ~~178~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a function call.

178 ~~179~~. A method as in claim 177 ~~178~~, in in which the contents of the function call are learned by executing steps (a) to (g).

179 ~~180~~. A method as in claim 177 ~~178~~, in which in which the contents

9

of the function call are not learned by executing steps (a) to (g).

180 ~~181~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises an external function call.

181 ~~182~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes the processor to execute a procedure in which the entire state of the processor is not saved before execution of the procedure.

182 ~~183~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes the processor to execute a leaf procedure.

RWS Q3

183 ~~184~~. A method as in claim 182 ~~183~~, ~~in~~ in which the contents of the leaf procedure are learned by executing steps (a) to (g).

184 ~~185~~. A method as in claim 182 ~~183~~, in which in which the contents of the leaf procedure are not learned by executing steps (a) to (g).

185 ~~186~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data to be retrieved from the memory.

186 ~~187~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data to be retrieved from the memory using an index.

187 ~~188~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes the processor to execute an instruction which precedes said directly executable instruction.

188 ~~189~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a loop instruction.

189 ~~190~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes the computer to execute in a recursive manner.

190 ~~191~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a list instruction.

191 ~~192~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a string instruction.

192 ~~193~~. A method as in claim 167 ~~168~~, in which said directly

16

executable instruction causes the processor to link the entity to a function which is external of the entity.

193 ~~194~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a protected division function.

194 ~~195~~. A method as in claim 167 ~~168~~, in which said directly executable instruction is a logarithmic function.

Q3

195 ~~196~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes the processor to execute, as a next instruction, an instruction other than an instruction which directly follows said instruction which directly follows said directly executable instruction in the entity.

196 ~~197~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data to be written to the memory.

197 ~~198~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data to be written to the memory using an index.

198 ~~199~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data in the memory to be altered.

199 ~~200~~. A method as in claim 167 ~~168~~, in which said directly executable instruction causes data in the memory to be altered using an index.

200 ~~201~~. A method as in claim 167 ~~168~~, in which said directly executable instruction comprises a control transfer instruction.

Sub B2 201 ~~202~~. A Turing complete computer implemented learning method comprising the steps of:

(a) providing a Turing complete computer with an indirectly executable program including:

a first instruction that points to and designates machine code stored in a memory as data;

a second instruction that points to and designates machine code stored in a memory as directly executable code;

a third instruction that alters machine code pointed to by the first instruction; and

~~a fourth instruction that executes machine code pointed to by the second instruction; and~~

~~(b) controlling the computer to execute the program which performs the steps of:~~

~~(b1) creating and storing a machine code entity including a directly executable instruction in a memory;~~

~~(b2) executing the second instruction to point to the entity;~~

~~(b3) executing the fourth instruction using input data to produce a result;~~

~~(b4) evaluating the result;~~

~~(b5) executing the first instruction to point to the entity;~~

~~(b6) executing the third instruction to alter the entity to include a different directly executable instruction; and~~

~~(b7) repeating steps (b3) to (b6) until an end criterion is reached.~~

202 ~~203. A~~ computer implemented learning method comprising the steps of:

(a) providing a computer with an indirectly executable program including:

a first instruction that points to and designates machine code stored in a memory as data;

a second instruction that points to and designates machine code stored in a memory as directly executable code;

a third instruction that alters machine code pointed to by the first instruction; and

a fourth instruction that executes machine code pointed to by the second instruction; and

(b) controlling the computer to execute the program which performs the steps of:

(b1) creating and storing a machine code entity including a directly executable instruction in a memory, said

[illegible]

directly executable instruction causing the processor to execute an instruction other than a next instruction which directly follows said directly executable instruction in the entity;

(b2) executing the second instruction to point to the entity;

(b3) executing the fourth instruction using input data to produce a result;

Q3 (b4) evaluating the result using a learning algorithm;

(b5) making a copy of the entity;

(b6) executing the first instruction to point to the copy of the entity;

(b7) executing the third instruction to alter the copy of the entity using an alteration algorithm to include a different directly executable instruction; and

(b8) repeating steps (b3) to (b7) until an end criterion is reached.

PLEASE AMEND THE SPECIFICATION AS FOLLOWS

Q4 Page 40, line 10: after "enabling" insert --routines including functions, main routines,--;

Pages 74 to 106 - delete these pages in their entirety.

Page 73, lines 6 to 11: delete the text in its entirety, and substitute therefor the following.

--The following articles were physically included in this application as originally filed, and constitute part of the Q5 original disclosure thereof.

1. COMPLEXITY COMPRESSION AND EVOLUTION, by Peter Nordin et al, Proceedings of the Sixth International Conference of Genetic Algorithms, Pittsburg, PA 1995.

2. GENETIC PROGRAMMING CONTROLLING A MINIATURE ROBOT, by Peter Nordin et al, Symposium on Genetic Programming, MIT, Nov. 1995.

9[illegible]